Highly Optimized Mathematical Functions for the Intel® Itanium® Architecture

Application Note

December 2004

Order Number: 245410-011

Contents

Figures

There are no figures in this version of the document.

Tables

1.0 Abstract

Intel Corporation is providing Intel® Itanium® assembler source code to evaluate certain core mathematical support functions for the C and FORTRAN programming languages. The intent is that these should replace less optimized implementations that would normally be provided by the compiler or OS vendor. The functions work well on the Itanium® 2 processor as well as the original Itanium® processor.

The present document explains the rationale behind this decision and summarizes important information on performance and accuracy of the Intel-provided functions. To download the source code, please visit http://developer.intel.com/software/products/opensource/libraries/num.htm.

2.0 Rationale

In programs that make extensive use of standard mathematical functions (e.g. sin, atan, exp, sqrt), overall performance can be significantly affected by the code used to evaluate these functions. Generally, compilers automatically link in, or sometimes in-line, standard code sequences provided by the hardware, compiler or OS vendor.

For portability, various standard algorithms coded up in C are commonly used, e.g. the FDLIBM (Freely Distributable LIBM) library.[1] However, to improve performance, Intel has provided a core set of the most important mathematical functions, hand-coded in Itanium assembler and carefully optimized, that are typically much faster than standard portable C functions. It is recommended that vendors replace any existing implementation of these core mathematical functions with the Intel-provided ones.

The Itanium architecture provides full support for the following three floating-point formats specified in the IEEE 754 floating-point Standard (IEEE 1985).

Table 1. Main Itanium® Processor Floating-point Formats

Format	Effective Precision	Total Bits in Encoding
Single	24	32
Double	53	64
Double-extended	64	80

For the core set of mathematical functions, separate variants are usually provided, with different names, that can be optimized for each precision. The standard convention is that the double-precision functions have no special suffix (e.g. “log”), single precision functions have an “f” suffix (e.g. “logf”) and double-extended precision functions have a “l” suffix (e.g. “logl”). These names are a de facto standard for C libm runtime libraries and are specified by the C99 Standard (ISO 1999).

The double precision functions provided by Intel have been aggressively tuned and optimized for performance without compromising accuracy. At present, a selection of the single precision functions provided are specially optimized. Others usually have the same performance as the double precision variants. Nevertheless, they are still typically faster than portable C versions that would otherwise be used, and provide almost perfect rounding.

1. See http://www.netlib.org/fdlibm.

The Intel-provided functions are accompanied by a configurable error handling routine, which defaults to C99-like behavior but may be configured to be compatible with FDLIBM error handling. The Intel error handler in its default configuration attempts to conform more closely to the IEEE 754 Standard for binary floating-point arithmetic (IEEE 1985).

3.0 Functions Provided

The functions provided by Intel are listed in Table 2. Note that this does not provide the full complement of functions that are usually available to programmers, e.g. all those mandated by the C99 standard. However, it is felt that this list contains most of the functions whose performance is normally important, and portable C versions can be used for the functions that are missing. In addition to C99 functions, Intel provides other functions including degree argument trigonometric functions, cotangent, sincos (returning both sine and cosine values), and financial functions annuity and compound.

Table 2. Mathematical Functions Currently Provided

Single	Double	Double-Extended
acosf	acos	acosl
acosdf	acosd	acosdl
acoshf	acosh	acoshl
annuityf	annuity	annuityl
asinf	asin	asinl
asindf	asind	asindl
asinhf	asinh	asinhl
atanf	atan	atanl
atan2f	atan2	atan2l
atan2df	atan2d	atan2dl
atandf	atand	atandl
atanhf	atanh	atanhl
cabsf	cabs	cabsl
cbrtf	cbrt	cbrtl
ceilf	ceil	ceill
compoundf	compound	compoundl
copysignf	copysign	copysignl
cosf	cos	cosl
cosdf	cosd	cosdl
coshf	cosh	coshl
cotf	cot	cotl
cotdf	cotd	cotdl
erff	erf	erfl
erfcf	erfc	erfcl
expf	exp	expl

Table 2. Mathematical Functions Currently Provided (Continued)

Single	Double	Double-Extended
exp10f	exp10	exp10l
exp2f	exp2	exp2l
expm1f	expm1	expm1l
fabsf	fabs	fabsl
fdimf	fdim	fdiml
floorf	floor	floorl
fmaf	fma	fmal
fmaxf	fmax	fmaxl
fminf	fmin	fminl
fmodf	fmod	fmodl
frexpf	frexp	frexpl
hypotf	hypot	hypotl
ilogbf	ilogb	ilogbl
invsqrtf	invsqrt	invsqrtl
isfinitef	isfinite	isfinitel
isgreaterf	isgreater	isgreaterl
isgreaterequalf	isgreaterequal	isgreaterequall
isinff	isinf	isinfl
islessf	isless	islessl
islessequalf	islessequal	islessequall
islessgreaterf	islessgreater	islessgreaterl
isnanf	isnan	isnanl
isunorderedf	isunordered	isunorderedl
ldexpf	ldexp	ldexpl
lgammaf	lgamma	lgammal
llrintf	llrint	llrintl
llroundf	llround	llroundl
logf	log	logl
log10f	log10	log10l
log1pf	log1p	log1pl
log2f	log2	log2l
logbf	logb	logbl
lrintf	lrint	lrintl
lroundf	lround	lroundl
modff	modf	modfl
nearbyintf	nearbyint	nearbyintl
nextafterf	nextafter	nextafterl
nexttowardf	nexttoward	nexttowardl

Table 2. Mathematical Functions Currently Provided (Continued)

Single	Double	Double-Extended
powf	pow	powl
remainderf	remainder	remainderl
remquof	remquo	remquol
rintf	rint	rintl
roundf	round	roundl
scalbf	scalb	scalbl
scalblnf	scalbln	scalblnl
scalbnf	scalbn	scalbnl
significandf	significand	significandl
sinf	sin	sinl
sincosf	sincos	sincosl
sincosdf	sincosd	sincosdl
sindf	sind	sindl
sinhf	sinh	sinhl
sinhcoshf	sinhcosh	sinhcoshl
sqrtf	sqrt	sqrtl
tanf	tan	tanl
tandf	tand	tandl
tanhf	tanh	tanhl
tgammaf	tgamma	tgammal
truncf	trunc	truncl

4.0 Speed

In Table 3, we give estimates of the number of clock cycles needed for an Itanium® 2 processor or Itanium® processor to execute one instance of each function.

Table 3. Approximate Timings for Functions Provided

Function	Single (f) Itanium 2 / Itanium	Double Itanium 2 / Itanium	Extended (l) Itanium 2 / Itanium
acos	39 / 43	46 / 50	93 / 108
acosd	40 / 43	46 / 50	78 / 91
acosh	52 / 66	67 / 84	117 / 133
annuity	84 / 98	103 / 123	203 / 230
asin	39 / 43	46 / 50	93 / 108
asind	40 / 43	46 / 50	78 / 91
asinh	52 / 66	67 / 84	113 / 129

Table 3. Approximate Timings for Functions Provided (Continued)

Function	Single (f) Itanium 2 / Itanium	Double Itanium 2 / Itanium	Extended (l) Itanium 2 / Itanium
atan	40 / 46	56 / 66	115 / 137
atan2	42 / 45	62 / 66	114 / 136
atan2d	45 / 45	60 / 66	83 / 95
atand	40 / 44	55 / 67	80 / 88
atanh	38 / 42	56 / 72	108 / 119
cabs	36 / 44	42 / 49	49 / 59
cbrt	31 / 37	34 / 40	42 / 51
ceil	13 / 18	13 / 18	13 / 18
compound	62 / 87	76 / 92	167 / 186
copysign	6 / 6	6 / 6	6 / 6
cos	42 / 49	49 / 62	84 / 90
cosd	38 / 38	47 / 56	68 / 77
cosh	41 / 48	48 / 58	67 / 79
cot	50 / 56	65 / 68	117 / 140
cotd	48 / 56	55 / 63	96 / 95
erf	35 / 37	50 / 53	72 / 78
erfc	53 / 53	68 / 65	104 / 109
exp	35 / 42	42 / 49	50 / 60
exp10	41 / 51	43 / 54	51 / 64
exp2	38 / 48	41 / 50	49 / 54
expm1	38 / 46	42 / 49	53 / 65
fabs	6 / 7	6 / 7	6 / 7
fdim	10 / 12	10 / 12	10 / 12
floor	13 / 18	13 / 18	13 / 18
fma	5 / 7	5 / 7	5 / 7
fmax	9 / 9	9 / 9	9 / 9
fmin	9 / 9	9 / 9	9 / 9
fmod	39 / 56	39 / 56	39 / 56
frexp	23 / 26	23 / 26	23 / 26
hypot	36 / 44	42 / 49	49 / 59
ilogb	8 / 5	8 / 5	8 / 5
invsqrt	25 / 31	30 / 37	32 / 38
isfinite	3 / 4	3 / 4	3 / 4
isgreater	3 / 4	3 / 4	3 / 4
isgreaterequal	3 / 4	3 / 4	3 / 4
isinf	3 / 4	3 / 4	3 / 4
isless	3 / 4	3 / 4	3 / 4

Table 3. Approximate Timings for Functions Provided (Continued)

Function	Single (f) Itanium 2 / Itanium	Double Itanium 2 / Itanium	Extended (l) Itanium 2 / Itanium
islessequal	3 / 4	3 / 4	3 / 4
islessgreater	3 / 4	3 / 4	3 / 4
isnan	3 / 4	3 / 4	3 / 4
isunordered	3 / 4	3 / 4	3 / 4
ldexp	16 / 24	16 / 24	16 / 24
lgamma	82 / 86	95 / 105	139 / 139
llrint	20 / 28	20 / 28	20 / 28
llround	20 / 28	20 / 28	20 / 28
log	27 / 29	31 / 35	68 / 69
log10	27 / 29	32 / 35	76 / 79
log1p	30 / 37	35 / 43	74 / 80
log2	32 / 36	36 / 41	46 / 51
logb	18 / 19	18 / 19	18 / 19
lrint	20 / 28	20 / 28	20 / 28
lround	20 / 28	20 / 28	20 / 28
modf	19 / 20	19 / 20	23 / 20
nearbyint	16 / 21	16 / 21	16 / 21
nextafter	25 / 25	25 / 25	25 / 25
nexttoward	25 / 25	25 / 25	25 / 25
pow	57 / 76	74 / 78	122 / 142
remainder	58 / 81	58 / 81	58 / 81
remquo	65 / 87	65 / 87	65 / 87
rint	16 / 21	16 / 21	16 / 21
round	19 / 27	19 / 27	19 / 27
scalb	21 / 34	21 / 34	21 / 34
scalbln	16 / 24	16 / 24	16 / 24
scalbn	16 / 24	16 / 24	16 / 24
significand	13 / 16	13 / 16	13 / 16
sin	43 / 49	49 / 62	83 / 90
sincos	48 / 57	59 / 70	88 / 98
sincosd	46 / 48	55 / 66	76 / 87
sind	38 / 38	47 / 56	69 / 78
sinh	41 / 48	48 / 58	67 / 79
sinhcosh	46 / 56	53 / 66	83 / 96
sqrt	32 / 42	40 / 51	44 / 56
tan	49 / 56	63 / 68	117 / 138
tand	48 / 56	55 / 63	96 / 95

Table 3. Approximate Timings for Functions Provided (Continued)

Function	Single (f) Itanium 2 / Itanium	Double Itanium 2 / Itanium	Extended (l) Itanium 2 / Itanium
tanh	33 / 37	51 / 53	72 / 78
tgamma	84 / 84	103 / 105	186 / 192
trunc	12 / 14	12 / 14	12 / 14

These cycle times are measured on the Itanium processors for what we consider the "most likely" path through the code. The measured times do not include overhead to call the function. The measurements assume that all pre-stored constants used internally are taken from second-level cache, which is only likely to be true when the functions are used quite frequently. In any case, actual times may be much less or much more depending on the particular function and argument. For example, the double precision sin function completes in only 19 cycles if the argument is zero, whereas it may take hundreds of cycles if the argument is very large ($>2^{63}$), necessitating a lengthier initial range-reduction step.

Generally speaking, the major double precision functions (e.g. sin, exp and pow) are 1.5 to 2 times faster than their double-extended precision counterparts, since many optimizations can be exploited when the final precision required is lower than the maximum supported by the architecture.

In addition, further optimizations have been made on major single precision functions (e.g. atan, sin, and exp). These are 1.2 to 1.6 times faster than their double precision counterparts. For other functions the single precision functions are simply the double precision ones with the final rounding changed to single precision. For those there is little or no performance gain from using the single precision functions in preference to the double precision ones.

5.0 Accuracy

Accuracy figures for a mathematical function may be obtained either by theoretical analysis to yield a proven error bound, or by extensive testing against a reliable higher-precision canon over a wide range of inputs. Both analysis and testing have been performed for some of the functions listed here. Table 4 gives maximum ulp (units in the last place) errors observed to date for some functions in round-to-nearest mode. Future testing may uncover larger errors.

Table 4. Accuracy Estimates for Functions Provided

Function	Single (f)	Double	Extended (l)
acos	0.506	0.503	0.517
acosd	0.505	0.502	0.530
acosh	0.501	0.508	0.539
annuity	0.501	0.504	0.546
asin	0.506	0.502	0.516
asind	0.506	0.502	0.530
asinh	0.508	0.504	0.514
atan	0.507	0.501	0.516

Table 4. Accuracy Estimates for Functions Provided (Continued)

Function	Single (f)	Double	Extended (l)
atan2	0.507	0.501	0.558
atan2d	0.507	0.502	0.519
atand	0.507	0.502	0.512
atanh	0.501	0.501	0.504
cabs	0.501	0.501	0.501
cbrt	0.503	0.501	0.503
compound	0.501	0.503	0.594
cos	0.502	0.503	0.556
cosd	0.501	0.502	0.515
cosh	0.501	0.501	0.534
cot	0.501	0.507	0.551
cotd	0.501	0.510	0.518
erf	0.513	0.504	0.515
erfc	0.501	0.502	0.526
exp	0.501	0.502	0.501
exp10	0.501	0.502	0.506
exp2	0.501	0.501	0.513
expm1	0.501	0.501	0.522
hypot	0.501	0.501	0.501
invsqrt	0.501	0.501	0.507
lgamma	0.509	0.509	0.544
log	0.501	0.501	0.511
log10	0.501	0.501	0.521
log1p	0.501	0.501	0.519
log2	0.501	0.501	0.503
pow	0.501	0.502	0.536
sin	0.502	0.503	0.565
sincos	0.502	0.503	0.565
sincosd	0.501	0.502	0.515
sind	0.501	0.502	0.515
sinh	0.501	0.502	0.534
sinhcosh	0.501	0.502	0.534
sqrt	0.500	0.500	0.500
tan	0.501	0.507	0.582
tand	0.501	0.510	0.518
tanh	0.510	0.502	0.515
tgamma	0.503	0.515	0.547

In general, a 0.5 ulp maximum error is the best we could expect, since this corresponds to always correctly rounding to the nearest floating-point number. It can be seen that the maximum errors observed are close enough to the ideal for most practical purposes. While these are not theoretical upper bounds, they are the *largest* errors encountered thus far in testing over the full range of input arguments, so in practice the observed errors will typically be lower.

Some other operations such as ceil, fabs, modf and remainder always return exact answers, so accuracy figures of this kind are not applicable. Moreover, the sqrt operation is provably always correctly rounded, and naturally no exception has been discovered by testing.

The double-extended functions may exhibit somewhat higher errors, but theoretical analysis suggests that the core functions are never in error by more than 0.59 ulp.

6.0 Further Reading

Story and Tang (1999) discuss the implementation of algorithms similar to the double-extended precision ones here, used by the Itanium processor to provide compatibility with IA-32 hardware transcendentals. Harrison, Kubaska, Story and Tang (1999) discuss how fast double precision functions were implemented; related techniques have be exploited to create faster single precision algorithms. For general discussion of implementing transcendental functions, see Muller (1997) and Markstein (2000). For discussion of the Itanium architecture, programming techniques, and applicability to scientific computing, see Cornea, Harrison and Tang (2002).

7.0 Authors

Yuri Akutin (Yuri.Akutin@intel.com)
Cristina Anderson (Cristina.S.Anderson@intel.com)
Marius Cornea (Marius.Cornea@intel.com)
Alexey Ershov (Alexey.Ershov@intel.com)
Eugeny Gladkov (Eugeny.Gladkov@intel.com)
Evgeny Gvozdev (Evgeny.Gvozdev@intel.com)
Bob Hanek (Bob.Hanek@intel.com)
John Harrison (johnh@ichips.intel.com)
Alexander Isaev (Alexander.Isaev@intel.com)
Andrey Kolesov (Andrey.Kolesov@intel.com)
Alexey Kovalev (Alexey.Kovalev@intel.com)
Elena Luneva (Elena.Luneva@intel.com)
Sergey Maidanov (Sergey.Maidanov@intel.com)
Andrey Naraikin (Andrey.Naraikin@intel.com)
Bob Norin (Bob.Norin@intel.com)
Pavel Shelepugin (Pavel.Shelepugin@intel.com)
Vladimir Sorokin (Vladimir.Sorokin@intel.com)
Shane Story (Shane.Story@intel.com)
Ping Tak Peter Tang (Peter.Tang@intel.com)

8.0 References

Cornea, M., Harrison, J., and Tang, P. (2002) *Scientific Computing on Itanium®-based Systems*. Intel Press.

Harrison, J., Kubaska, T., Story, S., and Tang, P. (1999) *The computation of transcendental functions on the IA-64 architecture*. Intel Technology Journal, 1999-Q4, 1–7. This paper is available on the Web as http://developer.intel.com/technology/itj/q41999/articles/art_5.htm.

IEEE (1985) *Standard for binary floating point arithmetic*. ANSI/IEEE Standard 754-1985, The Institute of Electrical and Electronic Engineers, Inc.

ISO (1999) *Programming languages - C*. International Standard ISO/IEC 9899:1999.

Markstein, P. (2000) *IA-64 and Elementary Functions: Speed and Precision*. Prentice Hall.

Muller, J.M. (1997) *Elementary functions: Algorithms and Implementation*. Birkhäuser.

Story, S. and Tang, P. T. P. (1999) *New algorithms for improved transcendental functions on IA-64*. 14th IEEE Symposium on Computer Arithmetic, pp. 4-11.

9.0 Revision History

Revision Number	Date	Description
1.0	January 2000	This is the first publication of this document.
2.0	April 2000	Updated Tables 2 and 3.
3.0	July 2000	Updated Table 3 to reflect improvements and measured timing. Updated Table 4 to add single and extended precision accuracy.
4.0	September 2000	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes.
5.0	November 2000	Updated trademark terminology
6.0	February 2001	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes.
7.0	June 2001	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes
8.0	February 2002	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes
9.0	December 2002	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes. Updated Table 3 to include Itanium 2 timing.
10.0	June 2003	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes.
11.0	December 2004	Updated Tables 2, 3, and 4 to reflect new functions and timing and accuracy changes.